Mainframe Interview Questions — JCL, COBOL & DB2
SECTION 1: JCL Interview Questions
SECTION 2: COBOL Interview Questions
SECTION 3: DB2 Interview Questions

Mainframe Interview Questions — JCL, COBOL & DB2

(Duplicates removed, reorganized by topic order: JCL → COBOL → DB2)

SECTION 1: JCL Interview Questions

1. What is the difference between SB37, SD37, SE37? - **SB37:** Abend occurs when the specified primary and secondary space is not enough. - **SD37:** Occurs when secondary space is not specified at all. - **SE37:** Same as SB37 (primary & secondary not enough), but occurs specifically for a PDS.

2. Explain the JOB card and its parameters (JOBNAME, positional & keyword parameters). `//JOBNAME POSITIONAL-PARAMETERS KEYWORD-PARAMETERS` - **JOBNAME:** 1–8 characters, alphanumeric/national (#,@,$), begins at column 3, no special characters. - **Positional parameters (mandatory):** Accounting information (max 142 chars) and Programmer name (max 20 chars, for documentation). - **Keyword parameters (can appear in any order):** - **CLASS:** Categorizes the job by CPU consumption/resource use. Range A–Z, 0–9. Jobs in the same class queue and execute one after another; different classes run in separate, simultaneous queues. - **PRTY:** Priority of execution, range 0–15 (higher = executed first). If jobs share class and priority and submit time, they wait until the operator intervenes. - **MSGCLASS:** Output device for messages (A–Z, 0–9), e.g., A=printer, X=spool. - **MSGLEVEL=(Statements, Messages):** Statements — 0=job messages only, 1=all JCL incl. PROCs/system msgs, 2=all JCL excl. PROCs/system msgs. Messages — 0=on success only, 1=on success or failure. Default is (1,1). - **NOTIFY=&SYSUID:** Sends success/failure notification to the specified user ID. - **REGION:** Work space for job/step execution, in KB/MB (REGION=0K/0M = max region). Insufficient region → abend S106. - **TIME=(minutes,seconds):** Max execution time for job/step. - **TYPRUN:** SCAN = check syntax only; HOLD = check syntax, hold job (release with line command 'A'); COPY = copy the job elsewhere. - **RESTART:** See Q6 below.

3. Maximum number of steps allowed in a job? Maximum 255 steps.

4. How do you avoid/bypass a positional parameter in the JOB statement? Use a comma to skip it before coding keyword parameters.

```
//JOBNAME JOB ',' NOTIFY=&SYSUID
```

5. How do you execute steps out of order, or execute only specific step(s), in a JCL with multiple steps? Two approaches: - **RESTART parameter** (job card): `RESTART=STEP7` starts execution from

STEP7 onward, bypassing earlier steps. - **IEBEDIT utility:** Lets you re-sequence or pick specific steps to submit via the internal reader (INTRDR), regardless of their physical order in the JCL.

```
//STEP1 EXEC PGM=IEBEDIT
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=MAC678.RANJITHA.PDS(PS1),DISP=SHR
//SYSUT2 DD SYSOUT=(*,INTRDR)
//SYSIN DD *
TYPE=EDIT, INCLUDE=(STEP2,STEP1,STEP3)
/*
```

To execute only step 3 and step 4: `INCLUDE=(STEP3,STEP4)`. To execute only step 5: `INCLUDE=(STEP5)`, or simply `RESTART=STEP5` if steps before it can be safely skipped.

6. Explain the RESTART parameter (job-level and PROC-level). - **Job-level:** `RESTART=STEP3` in the job card skips steps 1–2 and begins execution at STEP3 — useful after an abend so you don't rerun completed steps. - **PROC-level:** `RESTART=PROCNAME.STEPNAME` restarts from a specific step inside a called PROC.

7. What is the COND parameter, and how do you control step execution with it?

```
COND=(comparison-code, relational-operator, RC)
```

- If COND evaluates **TRUE → step is bypassed**. If **FALSE → step executes**.
- `COND=EVEN` — executes the step regardless of whether prior steps succeeded or failed.
- `COND=ONLY` — executes the step only if a prior step failed.
- `COND=(0,LE)` is always TRUE → step is **always bypassed**.
- `COND=(0,GT)` is always FALSE → step is **always executed**.
- To guarantee a step **never executes**, code `COND=(0,LE)` on it.
- COND values in an invoked PROC can be overridden from the calling JCL: `//STEP1 EXEC PROC,COND.STEPNAME-IN-PROC=NEW-VALUE`.

8. Can more than one job be submitted from a single JCL? How? Yes — using the **internal reader (INTRDR)**. One job's SYSUT2 writes to `SYSOUT=(*,INTRDR)`, and the SYSUT1 dataset contains the JCL of another job, which then gets submitted/executed via the internal reader.

9. How do you submit a job under a different user ID?

```
//JOB0001A JOB NOTIFY=&SYSUID,USER=OTHER-USERID,PASSWORD=PASSWORD-
OF-THE-USER
```

Useful, for example, when you lack access to a DB2 table but another user ID does.

10. How do you dynamically allocate/append to a dataset in JCL? Use `DISP=MOD`. If the dataset doesn't exist, it's created; if it exists, new records are appended at the end (as opposed to NEW which requires the dataset not exist, and OLD/SHR which don't append).

11. What are the DISP parameters? `DISP=(status, normal-termination, abnormal-termination)` - **Status:** NEW (doesn't exist, created here) / SHR (exists, shareable) / OLD (exists, exclusive — not shareable) / MOD (create if absent, else append). - **Normal termination:** CATLG (catalog it) / DELETE (no longer needed) / PASS (passed to later steps, usable once each) / KEEP (keep on volume) / UNCATLG (remove catalog entry). - **Abnormal termination:** CATLG / DELETE / KEEP / UNCATLG (**PASS is not allowed here**). - `DISP=(NEW,CATLG,CATLG)` — keeps/catalogs the output dataset even if the job abends after successful step completion (so you can still inspect output). - `DISP=(NEW,PASS,DELETE)` — dataset is temporary, passed to

later steps on success, deleted if the job abends. - **DISP=OLD vs DISP=SHR:** OLD explicitly locks the dataset (exclusive use); SHR allows multiple users/jobs to access it concurrently.

12. What is a Generation Data Group (GDG)? Explain creation, parameters, and referencing generations. A GDG is a group of datasets related to each other functionally/chronologically; each dataset is called a "generation." Used mainly for backups and reports — the same JCL can be reused without changing dataset names.

Creating a GDG base:

```
//STEP1 EXEC PGM=IDCAMS
//SYSIN DD *
DEFINE GDG(NAME(JOB001.RANJITHA.GDG)-
LIMIT(5)-
NOEMPTY/EMPTY-
SCRATCH/NOSCRATCH)
/*
```

- **NAME:** Dataset name of the GDG base.
- **LIMIT:** Max number of generations the GDG can hold (min 1, max 255).
- **EMPTY vs NOEMPTY** (mutually exclusive): EMPTY uncatalogs *all* generations once the limit is reached; NOEMPTY uncatalogs only the *oldest* generation.
- **SCRATCH vs NOSCRATCH** (mutually exclusive): SCRATCH physically deletes a removed generation; NOSCRATCH just uncatalogs it (data remains on volume).

Referencing generations: `(+1)` = new generation being created this run, `(0)` = current/most recent generation, `(-1)` = previous generation, `(-2)` = two generations back, etc.

Using a generation created earlier in the same job: Since GDG pointers only update at job end, a `(+1)` generation created in Step 1 must still be referenced as `(+1)` (not `(0)`) if used as input in Step 2 or Step 3 of the *same* job.

13. How do you check whether a file is empty, without opening it in a program?

```
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//INPUT DD DSN=JOB001.RANJITHA.INFILE,DISP=SHR
//SYSIN DD *
PRINT INFILE(INPUT) COUNT(1)
/*
```

RC=00 → file has data. RC=04 → file is empty.

14. How do you save/preserve a job's SYSOUT log for later reference? Type `XDC` in front of the job name in the SPOOL and press Enter, then supply a dataset name (must be defined with **LRECL=133, RECFM=VB**) to store the output.

15. How do you override a DD statement, or the dataset it points to, inside a PROC?

```
//stepname-in-proc.ddname-in-proc DD DSN=new.dataset.name,DISP=SHR
```

This works whether overriding a whole DD or just the DSN, and applies even without symbolic parameters.

16. What are Symbolic Parameters, and how are they used in a PROC? A symbolic is a placeholder in a cataloged PROC whose value is supplied at invocation time (`&symbol=value`); if not supplied, the PROC's coded default is used.

```
//PROC1 PROC UNIT=SYSDA        (default value of UNIT)
```

```
//S1 EXEC PGM=TEST1
//DD1 DD UNIT=&UNIT
//DD2 DD UNIT=&UNIT
```

Invoking `//STEP1 EXEC PROC1,UNIT=TEMPDA` overrides UNIT to TEMPDA for that run. Symbolics can be used in both EXEC and DD statements.

17. Difference between Instream and Cataloged procedures. How do you call a cataloged one? | Instream Procedure | Cataloged Procedure | |—|—| | Present in the same job stream | Stored in a separate PDS | | Statements between PROC and PEND | Also begins with PROC | | Max 15 invocations per job | Can be invoked unlimited times | | Called as `//STEP1 EXEC PROCNAME` | Called via `//JCLLIB ORDER=LOC.OF.DS` then `//STEP EXEC PROCNAME` |

18. Difference between JOBLIB and STEPLIB, and priority when both JOBCAT/STEPCAT exist. There are three library types: system library (SYS1.LINKLIB — no JOBLIB/STEPLIB needed for IBM utilities like IEFBR14, IEBGENER), private/user library, and temporary library. - **JOBLIB:** Applies to all steps in the job. - **STEPLIB:** Applies only to the step it's coded in — and **takes priority over JOBLIB** if both are present. - Similarly, if both STEPCAT and JOBCAT are coded, datasets are searched in **STEPCAT first**, then JOBCAT, then the system catalog.

19. How do you pass data from JCL to COBOL, and where do you declare it in COBOL? Two ways: - **PARM** (max 100 characters):

```
//S1 EXEC PGM=PROG1,PARM='4-1-2012'

LINKAGE SECTION.
01 LS-PARM.
   49 LS-PARM-LEN  PIC S9(4) COMP.
   49 LS-PARM-TEXT PIC X(100).
PROCEDURE DIVISION USING LS-PARM.
```

- **SYSIN DD *** — used when the data to pass exceeds 100 bytes.

20. What is the maximum length of the PARM parameter? 100 bytes.

21. `DCB=(LRECL=80,RECFM=FB,BLKSIZE=0)` — what happens when copying data with this? The dataset can be created (used as output) with BLKSIZE=0 (system determines optimal block size), but it **cannot be read back in** as an input dataset with this same undetermined blocksize expectation.

22. Max time limit settable in JCL, and how to specify a 45-second limit? - `TIME=1440` (hours), `TIME=248` (days), or `TIME=NOLIMIT`/`TIME=MAX` — all avoid the S322 time-limit abend. - For 45 seconds only: `TIME=(0,45)` or `TIME=(,45)`.

23. What is the use of RLSE in the SPACE parameter? It releases any unused (allocated but unwritten) space back to the system once the dataset is closed.

24. How do you use the SORT utility to copy from a PS file to a VSAM file?

```
//S1 EXEC PGM=SORT
//SORTIN DD DSN=FSS172.RANJI.PS1,DISP=SHR
//SORTOUT DD DSN=FSS172.RANJI.KSDS,DISP=SHR
//SYSIN DD *
SORT FIELDS=COPY
/*
```

25. How do you eliminate duplicate records using the SORT utility?

```
//S1 EXEC PGM=SORT
```

```
//SORTIN DD DSN=FSS045.RANJITHA.PS1,DISP=SHR
//SORTOUT DD DSN=FSS045.RANJITHA.PS2,DISP=SHR
//SYSIN DD *
SORT FIELDS=(1,10,CH,A)
SUM FIELDS=NONE
/*
```

26. A file has 100 records — how do you sort/keep only the first 40 records in the output?

```
//S1 EXEC PGM=SORT
//SORTIN DD DSN=JOB001.RANJITHA.PS1,DISP=SHR
//SORTOF01 DD DSN=JOB001.RANJITHA.PS2,DISP=SHR
//SYSIN DD *
 SORT FIELDS=(1,6,CH,A)
 OUTFIL FILES=01,STARTREC=01,ENDREC=40
/*
```

27. What is INREC and OUTREC? - **INREC:** Structure of records *before* sort is applied. - **OUTREC:** Structure of records *after* sorting (e.g., if input has 5 fields but you want only 3 in output, OUTREC defines that reduced layout).

28. How do you concatenate multiple files? - **IEBGENER** — concatenates up to 255 sequential (PS) files:

```
//STEP1 EXEC PGM=IEBGENER
//SYSUT1 DD DSN=FSS045.TEST.PS1,DISP=SHR,
//          DD DSN=FSS045.TEST.PS2,DISP=SHR
//SYSUT2 DD DSN=FSS045.TEST.PS3,DISP=SHR
//SYSIN DD DUMMY
```

- **IEBCOPY** — concatenates up to 15 PDS members/libraries:

```
//STEP1 EXEC PGM=IEBCOPY
//DD1 DD DSN=FSS045.TEST.PS1,DISP=SHR
//DD2 DD DSN=FSS045.TEST.PS2,DISP=SHR
//DD3 DD DSN=FSS045.TEST.PS3,DISP=SHR
//SYSIN DD *
COPY INDD=DD1,DD2,OUTDD=DD3
/*
```

29. How do you merge two sorted files?

```
//S1 EXEC PGM=SORT
//SORTIN1 DD DSN=FSS045.RANJITHA.PS300,DISP=SHR
//SORTIN2 DD DSN=FSS045.RANJITHA.PS400,DISP=SHR
//SORTOUT DD DSN=FSS045.RANJITHA.PS500,DISP=SHR
//SYSIN DD *
 MERGE FIELDS=(1,4,CH,A,1,4,CH,A)
/*
```

30. How do you rename a dataset? In ISPF, use line command `R` in front of the dataset name and supply the new name — the dataset is renamed automatically.

31. How do you check for JCL syntax errors without actually executing the job? `TYPRUN=SCAN` — validates JCL syntax only, without running any step.

32. What is a temporary dataset, and what happens to it across steps? A temp dataset (defined as `DSN=&&TEMPFILENAME`) exists only for the duration of the job and is auto-deleted once the job completes. If you have two temp datasets and jump directly to executing the second step, the first temp dataset remains available/unaffected within that same job run.

SECTION 2: COBOL Interview Questions

1. How do you resolve a U4038 abend? Two common causes: (1) mismatch between the logical file (COBOL SELECT/FD) and the physical file (JCL DD), or (2) STOP RUN is missing from the program.

2. A sequential file has 50 records — copy them into a new file in reverse order (last record first). - Option 1: Use File-Aid. - Option 2: `PERFORM VARYING WS-I FROM 50 BY -1 UNTIL WS-I = 0` — reading by relative record number in reverse and writing out.

3. How do you use parameters declared in the main program from within a subprogram? Via the **LINKAGE SECTION** in the subprogram, combined with the `USING` clause on the CALL statement (main) and `PROCEDURE DIVISION USING ...` (subprogram).

4. How do you define a file in COBOL and JCL?

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT INFILE ASSIGN TO DD1.

//DD1 DD DSN=DATASET-NAME,DISP=SHR
```

5. Difference between SYSOUT and SYSPRINT? - **SYSOUT:** Program's runtime output (after execution). - **SYSPRINT:** Compiler output — compilation messages/diagnostics.

6. How do you check whether a file is empty using the EMPTY option? The `EMPTY` clause/status check on the file lets you determine whether it contains any records without a full read loop.

7. How do you declare a dynamic (variable-size) array in COBOL?

```
WORKING-STORAGE SECTION.
05 WS-STUDENT OCCURS N TIMES ...
PROCEDURE DIVISION.
    ACCEPT N.
```

8. What are the types of CALLs? Difference between main and subprogram. - **Static call:** Subprogram name is a literal in the CALL statement; main and sub load modules are link-edited together at **compile time**. - **Dynamic call:** Subprogram name is held in a variable; the modules are linked at **execution time**. - **Main program:** Variables in WORKING-STORAGE SECTION; ends with STOP RUN. - **Subprogram:** Variables in LINKAGE SECTION; ends with GOBACK.

9. In a CALL, do the parameters need to match in type, length, and name? The **data type must match**; variable name and length can differ.

10. What is COMP-3 / packed decimal? How is memory and sign calculated? (COMP, COMP-1 through COMP-5) - **COMP:** Binary representation. S9(1)–S9(4) = half word (2 bytes); S9(5)–S9(9) = full word (4 bytes); S9(10)–S9(18) = double word (8 bytes). Most significant bit indicates sign. - **COMP-1:** Single-precision floating point, 4 bytes, no PIC clause. - **COMP-2:** Double-precision floating point, 8 bytes, no PIC clause. - **COMP-3 (packed decimal):** Each digit takes half a byte; sign is stored in the rightmost half-byte regardless of whether S is coded (C=positive, D=negative, F=unsigned positive). - Formula: **EVEN** number of digits → `(N/2) + 1` bytes. **ODD** number of digits → `(N+1)/2` bytes. - Example: `S9(5) COMP-3` (odd) = (5+1)/2 = **3 bytes**. `S9(7) COMP-3` (odd) = (7+1)/2 = **4 bytes**.

`S9(5) COMP` = **4 bytes** (full word). - **COMP-4:** Same as binary/COMP — values may truncate depending on compiler options. - **COMP-5:** Binary integer, but **never truncated**. - For high-precision floating point, **COMP-2** is preferred over COMP-1 (8 bytes vs 4 bytes precision).

11. How do you print/display a COMP-3 value in a report? Use the `HEX ON` command in the file viewer, or move the COMP-3 value to an intermediate display-format variable (e.g., `PIC 9(4).9(2)`) before displaying/printing it.

12. A file already exists with a fixed record length — can you extend that length? No — the DCB (record length) parameters of an existing dataset cannot be changed once defined; you'd need to recreate the dataset with new attributes.

13. What is a clustering index? Every VSAM (KSDS) file has an associated clustering index that physically organizes records by key.

14. How do you pass data from COBOL back to JCL? - `MOVE 4 TO RETURN-CODE` — sets the program's return code, checkable via COND in later JCL steps. - Through files — writing data that subsequent JCL steps' DD statements reference.

15. What does File Status 92 mean? A logical error — occurs when a file opened in **INPUT** mode is then subjected to a WRITE/REWRITE/DELETE operation (a mode mismatch).

16. What is the EVALUATE statement, and how do you code compound (AND) conditions? Used to replace nested-IFs for cleaner conditional logic.

```
EVALUATE TRUE
    WHEN COND-1
        DISPLAY 'COND-1'
    WHEN A = 1 AND B = 2
        DISPLAY 'B=2'
    WHEN OTHER
        DISPLAY 'OTHERS'
END-EVALUATE.
```

- Max 255 WHEN conditions; WHEN OTHER must be last (always true).
- THRU/THROUGH specifies a range of values.
- END-EVALUATE is a mandatory scope terminator.

17. SEARCH vs SEARCH ALL — which is better for large tables? | SEARCH | SEARCH ALL | |—|—| | Linear search | Binary search | | Checks elements one by one | Repeatedly halves the array to locate the value | | Data need not be sorted | Data **must** be sorted | | Multiple WHEN conditions allowed | Only one WHEN condition allowed | | Any relational operator (LE, GT, GE, EQ, NE) | Only = operator | **SEARCH ALL is more efficient for large tables** (binary search vs linear).

18. Which level number is used for condition names? Level 88.

19. How many divisions are there in COBOL? Four: Identification, Environment, Data, Procedure.

20. Static vs Dynamic call — compiler options? Static call → `PARM=NODYNAM`. Dynamic call → `PARM=DYNAM`, specified at compile/link time.

21. What is REDEFINES? What are the rules? Allows a data item to share the same memory as another.

```
01 NAME1 PIC A(10) VALUE 'ARUN KUMAR'.
01 NAME  REDEFINES NAME1.
   02 FIRST-NAME PIC A(5).
   02 LAST-NAME  PIC A(5).
```

- REDEFINES is valid: elementary-to-elementary, group-to-group.
- REDEFINES is **not valid** for sub-elementary to sub-elementary items.
- The level numbers of the redefining and redefined items must be the **same**.
- **You cannot REDEFINE a copybook.**
- Common use case: validating one field's data by reinterpreting it as another layout.

22. What is REFERENCE MODIFICATION, and what are the ways to split strings? Four techniques: **RENAMES**, **REDEFINES**, **REFERENCE MODIFICATION**, **UNSTRING**. Reference modification moves a substring from position:length —

```
01 A PIC A(10) VALUE 'ARUN KUMAR'.
01 B PIC A(5).
01 C PIC A(5).
PROCEDURE DIVISION.
    MOVE A(1:4) TO B.
    MOVE A(6:5) TO C.
```

23. How do you count total characters in a string?

```
INSPECT VARIABLE-NAME1 TALLYING WS-CNT FOR ALL CHARACTERS.
```

24. How do you count total records in a file?

```
PERFORM UNTIL WS-EOF = 'Y'
    READ INFILE
        AT END MOVE 'Y' TO WS-EOF
        NOT AT END ADD 1 TO WS-CNT
    END-READ
END-PERFORM.
CLOSE INFILE.
DISPLAY WS-CNT.
```

25. How do you copy only even-numbered records to an output file?

```
PERFORM UNTIL WS-EOF = 'Y'
    READ INFILE
        AT END MOVE 'Y' TO WS-EOF
        NOT AT END
            ADD 1 TO WS-CNT
            IF WS-CNT = 2
                MOVE IN-REC TO OUT-REC
                WRITE OUT-REC
                MOVE 0 TO WS-CNT
            END-IF
    END-READ
END-PERFORM.
```

26. Can you delete records from an ESDS file? No — DELETE is not possible on an ESDS. You can only add records at the end; existing records cannot be updated in place (content-wise) or deleted.

27. What is SSRANGE/NOSSRANGE? Difference between INDEX and SUBSCRIPT. | Subscript | Index | |—|—| | Occurrence number of an array element | Displacement value of an array element | | Declared as a separate WORKING-STORAGE variable | Declared via INDEXED BY clause | | Incremented/decremented via arithmetic (ADD/SUBTRACT) | Incremented/decremented via SET | | Slower access | Faster access | - **SSRANGE:** Enables subscript range checking (used for dynamic arrays). `PARM=SSRANGE`. - **NOSSRANGE:** No range checking (default) — `PARM=NOSSRANGE`, typically for static arrays.

28. How do you identify whether a CALL is static or dynamic, just by looking at the code? `CALL 'PGM-NAME'` (literal in quotes) → **static**. `CALL VARIABLE-NAME` (a data item) → **dynamic**.

29. What are the disadvantages of VSAM files? - Requires significant DASD space. - For KSDS, the primary key cannot be changed once set. - Performance can degrade due to index complexity. - VSAM files reside only on DISK (not tape).

30. How do you know which type of VSAM file (ESDS/KSDS/RRDS) you're using, from the code? In FILE-CONTROL's SELECT clause: - `ORGANIZATION IS SEQUENTIAL` → ESDS - `ORGANIZATION IS INDEXED` → KSDS - `ORGANIZATION IS RELATIVE` → RRDS - (with `ACCESS MODE IS ...` and `RECORD KEY IS ...` for KSDS)

31. Is COBOL an object-oriented or structured programming language? Traditionally **structured**; VS COBOL II / Enterprise COBOL introduced some OO extensions in later versions.

32. Which COBOL compiler version is commonly referenced? Enterprise COBOL / VS COBOL II (e.g., V3.1).

33. Difference between NEXT SENTENCE and CONTINUE? - **CONTINUE:** Passes control to the statement immediately following the last scope terminator. - **NEXT SENTENCE:** Passes control to the statement immediately after the next period (`.`) — which can skip more logic than intended, so it's used carefully.

34. What are the data types in COBOL? - **Numeric:** up to 18 digits. - **Alphabetic & Alphanumeric:** up to 32,767 bytes.

35. How do you declare an array?

```
01 I     PIC 9 VALUE 1.
01 ARRAY.
   05 ARRAY1 PIC X(9)  OCCURS 10 TIMES.
   05 ARRAY2 PIC X(20) OCCURS 20 TIMES INDEXED BY I.
```

36. What is a copybook, and how do you use it (including REPLACING)? A copybook stores a predefined, reusable record/data structure that gets physically copied into the source at compile time via the `COPY` statement.

```
ENVIRONMENT DIVISION.
    COPY WORKCOPY.
```

To reuse the same copybook for two different field names:

```
COPY MEM1 REPLACING == EMP-ID == BY == EMP-NO ==.
```

37. What does EXIT do in COBOL? `EXIT` performs no operation by itself — it's typically used as a placeholder/landing point (e.g., at the end of a PERFORM'd paragraph).

38. What is a NESTED-IF statement? An IF statement containing another IF as one of its branches. There's theoretically no depth limit, but nested IFs should be used sparingly — EVALUATE is usually clearer when testing a variable against many values.

```
IF CONDITION-1
    IF CONDITION-2
        STATEMENT-1
    ELSE
        STATEMENT-2
    END-IF
ELSE
    STATEMENT-3
END-IF.
```

39. What error/abend codes are commonly seen in COBOL, and what causes each? - **SOC7 (data exception):** Invalid/non-numeric data in a numeric field — either a bad MOVE, or non-numeric junk present in a field declared as numeric in the file. **Resolution:** In the SPOOL, view CESDUMP to get the offset address, locate the

corresponding line number in SYSPRINT, then navigate to that line in the source and fix the offending data/logic. - **SOC4 (storage violation):** Missing SELECT statement at compile time, a bad subscript/index, attempting read/write on an unopened file, or missing parameters in a called subprogram. - **U4038:** Logical/physical file mismatch, or missing STOP RUN. - Other known codes: **SOCB, SOC1**.

40. GOBACK vs STOP RUN — what happens if you use GOBACK instead of STOP RUN? - **STOP RUN:** Used in the *main/calling* program — returns control to the operating system. - **GOBACK:** Used in a *called subprogram* — returns control to the calling program. - If STOP RUN is mistakenly coded in a called subprogram, it can end the entire job/run prematurely instead of returning to the caller (potential infinite loop or premature termination depending on context) — GOBACK should be used in modules that are always called by another program.

41. Truncation examples on MOVE (alphanumeric and numeric):

```
01 A PIC X(5).
01 B PIC X(3).
MOVE 'ABCDE' TO B.   *> B = 'ABC' (leftmost 3 chars retained)

01 A PIC 9(5).
01 B PIC 9(3).
MOVE 12345 TO B.     *> B = 345 (rightmost 3 digits retained)
```

SECTION 3: DB2 Interview Questions

1. A DB2 table has 1000 rows. After updating the first 110 rows, the job abends — how do you restart from row 111? Use a WORKING-STORAGE counter with periodic COMMITs so you have checkpoint boundaries to resume from, rather than restarting the whole batch:

```
EXEC SQL INCLUDE SQLCA END-EXEC.
EXEC SQL INCLUDE DCLGEN1 END-EXEC.

PROCEDURE DIVISION.
    EVALUATE SQLCODE
        WHEN 0
            COMPUTE ESAL = ESAL * 110 / 100
            EXEC SQL
                UPDATE RANJITHA1 SET ESAL = :ESAL
            END-EXEC
            ADD 1 TO WS-CNT
            IF WS-CNT = 10
                EXEC SQL COMMIT END-EXEC
                MOVE 0 TO WS-CNT
            END-IF
        WHEN 100
            CONTINUE
    END-EVALUATE.
```

Committing every N rows means an abend only loses uncommitted work since the last COMMIT, so restart logic just needs to pick up from the last committed point.

2. How do you identify a COBOL-DB2 program just by looking at the JCL? It runs under `PGM=IKJEFT01` (TSO batch utility) with `PARM(PARM-NAME,PLAN-NAME)` in SYSTSIN — the presence of a DB2 **PLAN** is the giveaway.

3. How do you pass data from JCL to a COBOL-DB2 program (since PARM= on EXEC is used for plain COBOL)? At run time, via the **SYSTSIN card** of the `IKJEFT01` step, using its PARM parameter.

4. Explain Precompilation, Timestamp Mismatch (-818), and Null Indicators. - **Precompilation:** The COBOL-DB2 source is first fed to the DB2 **precompiler**, which extracts embedded SQL into a **DBRM** and replaces those SQL statements with COBOL CALL statements. This modified source then goes through the normal COBOL compiler and link-editor to produce the load module. A **timestamp token** is stamped on both the modified source and the DBRM during this step. - **SQLCODE -818 (Timestamp Mismatch):** A runtime error caused when the timestamp in the DBRM (bound into the plan/package) doesn't match the timestamp in the load module (DBRMLIB vs LOADLIB out of sync). **Fix: REBIND the program.** - **Null Indicator:** A separate host variable used to detect NULLs when fetching DB2 data into COBOL fields (COBOL has no native NULL concept). - Indicator **< 0** → the column value is NULL (COBOL variable holds a meaningless default based on its data type). - Indicator **= 0** → value was retrieved properly, not null. - Indicator **> 0** → value was retrieved but got **truncated**. - Declared as: `01 EMP-LANDLINE-NO PIC S9(4) COMP.` — this indicator variable must exist in WORKING-STORAGE to both detect and *set* a column to NULL (e.g., moving -1 to it before an INSERT/UPDATE).

5. What is a cursor in DB2? Explain the steps, and what causes -811. A cursor is required whenever a SELECT can return **more than one row**. Four steps: 1. **DECLARE** the cursor 2. **OPEN** the cursor 3. **FETCH** rows from the cursor (in a loop) 4. **CLOSE** the cursor

```
EXEC SQL
    DECLARE CURSORNAME CURSOR FOR
    SELECT * FROM TABLENAME
END-EXEC.
```

To keep a cursor open across a COMMIT (normally COMMIT auto-closes cursors), declare it `WITH HOLD`:

```
EXEC SQL
    DECLARE CURSORNAME CURSOR WITH HOLD FOR
    SELECT * FROM TABLENAME
END-EXEC.
```

- A cursor can be declared in either the WORKING-STORAGE SECTION or the PROCEDURE DIVISION.
- **SQLCODE -811** occurs when a program tries to retrieve **multiple rows via a singleton SELECT INTO** (without a cursor). Fix: either use a cursor, or restrict the WHERE clause to guarantee one row, or use `SELECT * FROM TABLE FETCH FIRST n ROWS ONLY` to sequentially cap/retrieve rows without a full cursor loop.
- Forgetting to **CLOSE** the cursor can also cause problems — e.g., a table opened in batch mode with an open cursor cannot then be opened for the same use in an online (CICS) region until it's closed.

6. What are DB2 Isolation Levels? Three levels, specifying the degree of locking: 1. **Cursor Stability (CS):** Row-level locking — an exclusive lock (X-lock) is acquired on the row being updated and released once the cursor moves to the next updatable row. 2. **Repeatable Read (RR):** Page-level locking — an X-lock is acquired on the entire page containing the row(s), released only when control moves to the next updatable page. 3. **Uncommitted Read (UR):** Reads data even if it's been modified but not yet committed by another process (a U-lock is used) — fastest, but can read "dirty" data.

7. Common DB2 SQLCODEs and what causes them: | SQLCODE | Cause | Typical fix | |—|—|—| | **-805** | DBRM/Package not found in the PLAN | REBIND/re-add the package to the plan | | **-811** | Multiple rows returned from a singleton SELECT (no cursor used) | Use a cursor, or a more selective WHERE, or FETCH FIRST n ROWS ONLY | | **-818** | Timestamp mismatch between DBRMLIB and LOADLIB | REBIND the program | | **-911** | Deadlock/timeout — resource held by another process | Retry after a delay | | **-922** | Authorization failure | Request table access from the DBA |

8. How do you fetch just the first row (or first N rows) of a table?

```
SELECT * FROM TABLE FETCH FIRST 1 ROWS ONLY;    -- first row
SELECT * FROM TABLE FETCH FIRST n ROWS ONLY;    -- first N rows
```

9. How do you select rows with NULL data in a column?

```
SELECT * FROM TABLE WHERE COLNAME IS NULL;
```

(Note: `= NULL` is not valid SQL for null comparison — `IS NULL` is required.)

10. How do you retrieve a substring in DB2 SQL?

```
SELECT SUBSTRING(FIELDNAME, START-POSITION, LENGTH) FROM TABLE;
```

11. How do you insert a NULL value into a specific column?

```
INSERT INTO TABLENAME VALUES ('A001', 'RAHUL', NULL);
```

In COBOL, this requires a **null indicator variable** (see Q4) set to a negative value before the INSERT/UPDATE.

12. Difference between Primary Key and Unique Index. - **Primary Key:** Enforces NOT NULL and uniqueness on a column (no duplicates, no nulls) — uniquely identifies each row. - **Unique Index:** The mechanism DB2 uses to physically enforce and speed up lookups on that primary key — a primary key **cannot exist without an underlying unique index**.

13. What is DCLGEN? Is it mandatory? What does it contain? **DCLGEN (Declaration Generator)** is a DB2 tool that generates COBOL-equivalent host variable declarations matching a DB2 table's column names and data types — reducing manual data-type mismatches between DB2 and COBOL. - It contains: DB2 column names & data types, corresponding COBOL field names, COBOL data types, and lengths. - **DCLGEN is optional** — you *can* manually declare host variables in COBOL-equivalent types, but you risk type mismatches; DCLGEN removes that risk. - Used in a program as:

```
EXEC SQL
    INCLUDE DCLGEN-MEMBER-NAME
END-EXEC.
```

14. When and how do you REBIND a program? REBIND when the SQL statements themselves haven't changed but the **access path may benefit from being recalculated** — e.g., a new index was added, or RUNSTATS was recently run. REBIND is cheaper than a full BIND with `ACTION(REPLACE)`.

```
//SYSTSIN DD *
REBIND PLAN(PLAN-NAME) VALIDATE(BIND)
```

15. Explain the Precompilation → Bind → Execution sequence for a COBOL-DB2 program. 1. **Precompilation:** Utility **DSNHPC**. Extracts SQL into a DBRM, replaces SQL in source with COBOL CALLs, stamps a timestamp on both. 2. **Compile & Link-edit:** Standard COBOL compiler + linkage editor produce the load module. 3. **Bind:** Utility **IKJEFT01**. Converts DBRM(s) into a **PACKAGE**

and/or **PLAN** with an access path (see Q19). 4. **Execution:** Program runs under the bound PLAN, with the DB2 subsystem/region, program name, and PLAN name specified in the RUN JCL (via PARM in SYSTSIN).

16. Difference between INCLUDE and COPY. Both copy statements from a member into the main program at build time — functionally similar, but: - **INCLUDE** is resolved during **precompilation** (used for DB2 members like SQLCA, DCLGEN). - **COPY** is resolved during **compilation** (standard COBOL copybooks).

17. What is COMMIT? Saves all changes made to the database **permanently** since the last COMMIT (or job start). On successful program completion, an auto-COMMIT is issued; on unsuccessful completion, an auto-ROLLBACK is issued instead.

18. Explain the BIND process and its key parameters. What's the difference between a PLAN and a PACKAGE?

```
//SYSTSIN DD *
DSN SYSTEM(DB9G) -
BIND PACKAGE(PACKAGE-NAME) MEMBER(MEMBER-NAME) -
     OWNER(OWNER-NAME) QUALIFIER(QUALIFIER-NAME) -
     VALIDATE(BIND) ISOLATION(CS) -
     RELEASE(COMMIT) ACTION(REPLACE) -
     EXPLAIN(NO)
END
```

- **SYSTEM:** Target DB2 subsystem/region.
- **PACKAGE:** Name under which the DBRM + access path is bound.
- **MEMBER:** The COBOL-DB2 program's member name.
- **OWNER / QUALIFIER:** Table owner, and the qualifier used to distinguish test vs production tables (typically the same in production, different in dev/test).
- **VALIDATE:** Checks authorization for binding/execution.
- **RELEASE(COMMIT):** Locks release at COMMIT; **RELEASE(DEALLOCATE)** releases only when the plan/package deallocates.
- **ACTION:** ADD (new package, added to SYSIBM.SYSPACKAGE) or REPLACE (overwrite existing).
- **EXPLAIN(YES):** Produces the best access path info (typically paired with RUNSTATS output).
- **PLAN vs PACKAGE:** One DBRM is generated per program. If a main program calls several subprograms, each gets its own DBRM. These DBRMs can be bound directly into a single executable **PLAN**, or first grouped into intermediate **PACKAGES** (one or more DBRMs each), which are then bound into the PLAN. A **package is not directly executable**; a **PLAN is required to run** a DB2 program.

19. How do you get the row count of a table?

```
SELECT COUNT(*) FROM TABLENAME;
```

20. How do you retrieve only unique/distinct records?

```
SELECT DISTINCT column-list FROM TABLENAME;
```

21. How do you find duplicate records when you don't know where they are?

```
SELECT CLASS, SECTION, COUNT(*) FROM EMP_TBL
GROUP BY CLASS, SECTION
HAVING COUNT(*) > 1;
```

22. Can you insert multiple records at once? What is SPUFI? - Multiple-row inserts can be done via **dynamic SQL**. - **SPUFI (SQL Processing Using File Input)** — an interactive DB2 tool to run ad-

hoc SQL. In SPUFI, terminate each statement with ; to run several statements in one pass; prefix a statement with - to skip executing it.

23. Can you select from more than one table at once?

```
SELECT * FROM TABLE1, TABLE2;
```

(An implicit join — typically paired with a WHERE clause to relate the tables.)

24. Can a table have more than one primary key? No — a table can have only **one** primary key (though that key can be composite, spanning multiple columns).

25. What is a tablespace, and what types exist? A tablespace is the storage area for tables/views. Three types: 1. **Simple:** A page can hold data from more than one table. 2. **Segmented:** A page holds data from only a single table. 3. **Partitioned:** Data is split into groups of pages by partition — used for very large tables.

26. What is RDBMS? Any other RDBMS you know? RDBMS = Relational Database Management System — data is organized into tables with defined relationships between columns (e.g., via keys). (Other common RDBMS: Oracle, SQL Server, MySQL, PostgreSQL.)

27. What utilities are used to run a COBOL-DB2 program? - **DSNHPC** — precompilation. - **IKJEFT01** — bind and execution (TSO batch).

28. What operations happen when compiling a COBOL-DB2 program? Three stages: **1) Precompilation → 2) Bind → 3) Execution** (see Q15 for full detail).

29. What is EXPLAIN in DB2? `EXPLAIN(YES)` during BIND causes DB2 to generate the **best access path** (using RUNSTATS statistics) and store it for the package; `EXPLAIN(NO)` skips this.

30. What is the SQL Communication Area (SQLCA)? The communication area between COBOL and DB2 that reports the status of the last SQL statement executed.

```
EXEC SQL
    INCLUDE SQLCA
END-EXEC.
```

Resolved at precompile time into a structure containing SQLCODE, SQLSTATE, SQLERRM, SQLWARN, etc. - **SQLCODE = 0** → successful execution, row found. - **SQLCODE = +100** → successful execution, but **no row found**. - **SQLCODE < 0** → unsuccessful execution (an error occurred).

31. What do you specify in the RUN JCL for a COBOL-DB2 program? The DB2 subsystem/region name, the program to run, the DB2 **PLAN** name, and PARM data if the program needs it — all typically supplied via the SYSTSIN card under an `IKJEFT01` step.

Note: The original document also included CICS and IMS-DB sections, which were excluded here per the requested JCL → COBOL → DB2 scope. Let me know if you'd like those cleaned up and added as well.